This column focuses on secure computing, providing tools and tips for those in the information security trenches. Each issue, we'll evaluate new technologies (primarily in the open source space) and discuss ways to integrate them into your organization.

We want to hear from you. Got a great utility or "magic" script that's saved you hours of tedious keyboard pounding? Something new we haven't heard about? Let us know at ciocorner@sandstorm.net.

Knock Knock

"Security through obscurity" has gotten a bad rap in the infosec realm. Basically it means that "security" is achieved by concealing protection mechanisms under the assumption that no one will be able to figure out how to break them. In many cases, this reasoning is demonstrably bad, and recognized as such by the community. However one must not summarily discount applications because they fall into this category. A little common sense goes a long way.

A good example is "port knocking," a cleaver alternative to firewall authentication. The standard approach to firewalling is to deny or allow access to certain hosts or applications based on the IP address of the client machine. This works well for remote access if the client IP is known ahead of time, but the majority of broadband and dial-up users are assigned dynamic addresses. Also, users may need to connect on the road (an Internet cafe, for example). Port knocking tackles this problem by requiring a client to send a series of "knocks" before they are allowed to connect. The cleaver bit is that the knocks are sent to closed TCP ports.

At its core, this may seem like security through obscurity, and be dismissed as such. Sure, it's obscuring access points and adding another layer of complexity to authentication. But what's wrong with that? There are arguments on both sides of the issue (and an excellent rebuttal by Jay Beale, author of one port knocking implementation at www.port-knocking.org). In reality, port knocking is an authentication scheme that necessarily obscures. It makes great sense when you want to protect stuff.

One implementation is available at www.portknocking.org. The reference system uses the logging capabilities of IP tables to verify authentication. A client sends a series of connection attempts (SYN packets) on various ports to a firewall. The packets are silently dropped and logged. A port knocking daemon (running a machine with access to your firewall) parses these logs looking for the correct series of connection attempts. Once detected, it can dynamically modify the firewall rules to allow temporary access to any port from a specified IP address.

Given enough time and enough Perl, you could use a port knocking sequence to start the coffee machine at work. As Beale notes, SSH is an excellent application to protect with port knocking. Obviously this system doesn't work with public services such as web or email, but does with SSH and its ilk that are access-limited to trusted users. With SSH, port 22 would appear closed until the client completed the correct knock, at which point it would be opened for a brief (configurable) period.

The benefit of this system is that it provides remote access from arbitrary clients, while at the same time obscuring the fact that it is providing service. Since the service port is closed, it is likely to be passed over by worms or "random" Internet scans. This may afford the administrator a little more time to patch a vulnerable system when an exploit is discovered.

PORT KNOCKING SYSTEMS

A primitive port knocking system might work like this: A client sends SYN packets to ports 5643, 234, 6543, 6679, 35546, 192, 14 to an IP Tables firewall. IP Tables detects and logs the connection attempts. The port knocking daemon notices the log file has changed, and looks for the correct sequence. Seeing that the correct sequence was entered, it modifies the IP Tables ruleset allowing access to SSH.

A simple knock sequence is easy to set up, but to harden the system, cryptography must be incorporated. Since network traffic may be intercepted, knocks must not be reproducible, as an obvious attack method would be to watch a successful connection and simply replay it. While the reference system provides for encrypted knocks, it is not intended for use in a production environment.

Because the system uses SYN packets to relay data, there is far more overhead than standard data channels. Passing sufficient encrypted data with knocks hurts efficiency. More work needs to be done in this area to balance security and usability.

Another potential problem with the system is that while it does not require direct modification of server software, additional client software is required to securely construct the knock sequence. Clients must be able to initiate connections on arbitrary TCP ports. Either of these functions may be forbidden in certain computing environments. Clients must also be informed of the correct knock sequences ahead of time, since passing this data over the same channel as the authentication is principally bad.

This isn't a new idea. The "black hat" community has been using similar techniques to hide covert entry points on compromised systems for years. This may be a case where the evil uses far outweigh the good ones (i.e., it might be a lot more useful to the hacker hiding his root shell from your port-scanner than for you to obscure service from your users).

KNOCKING FUTURE

However, the recent rise in popularity and community awareness around port knocking may lead to a more stable (and ultimately more useful) implementation. We're not there yet. But with the reference code, an IP Tables box, your favorite scripting environment and a little imagination, port knocking can be molded in some really interesting ways. It's not for everyone, but likely a welcome addition to certain computing niches. For example, creating your own Knoppix disc that mounts a persistent home directory over the Internet.

In this example, a custom Knoppix distribution would be constructed containing a port knocking client. This client would be configured to initiate a knock on a remote system during startup to open SSH service. An NFS export could then be tunneled through an SSH connection to mount a remote home directory that would persist after the Knoppix session ends. Public/private key encryption could be incorporated using a USB key chain. This would provide a complete, secure and extremely portable system (using free software).

Summarily dismissing a creative application like port knocking because it seems "obscure" has the potential to let a clever way to harden a firewall go by the wayside. Imagine a day when all the ports are closed on all the firewalls. Certainly this would put a lid on the current port-scanning problem. CDM

RESOURCES:
http://www.portknocking.org/

http://www.linuxjournal.com/

http://www.netfilter.org/

http://www.knoppix.org/

Walker Whitehouse *is CIO and* **Mike Yamamoto** *is a Network Systems Engineer at Sandstorm Enterprises, which develops aggressive software products for network monitoring, network forensics analysis, and security auditing including telephone scanning, penetration testing, and vulnerability assessment.*